

**National Center for Atmospheric Research (NCAR)
High Altitude Observatory (HAO)
Boulder, CO.**

**OPeNDAP Server for the Earth System Grid (ESG)
User's Manual**

This manual will guide you through the usage of the OPeNDAP Server and Client for the Earth System Grid. It contains how to start and stop the server, how to start the client, and what commands are available for use within ESG. We will also show you some of the different configuration options available in our configuration file.

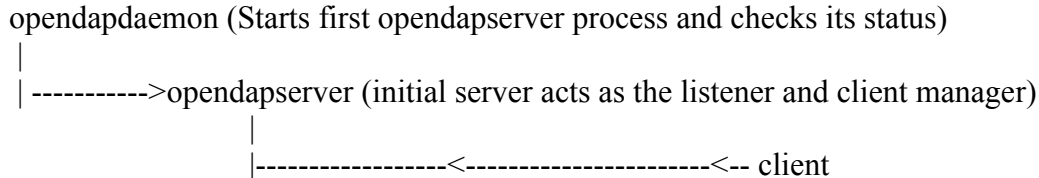
Patrick West
pwest@hao.ucar.edu

1. The OPeNDAP Server 4 Server Architecture

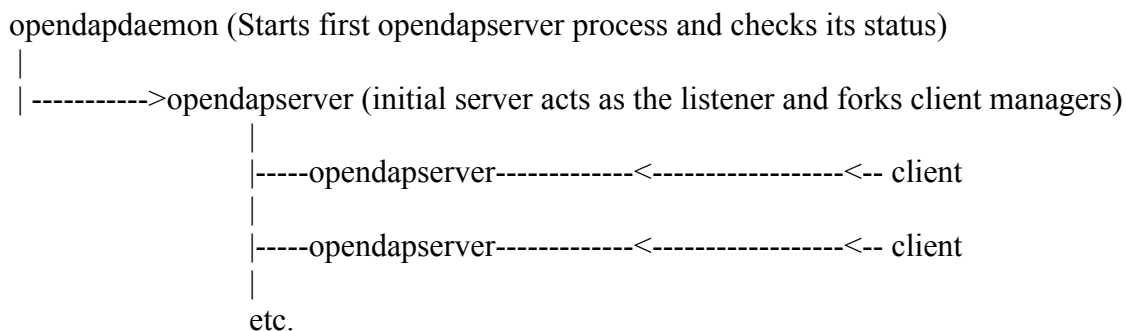
The OPeNDAP Server 4 server is a reliable back-end server which allows for either a single client to connect at a time or for multiple clients to connect at the same time, depending on the configuration of the system you choose to have. Each client connection is provided with its own process space in order to provide the maximum performance and reliability of the system.

For this to happen we have designed the system to first start a daemon process that will ensure that there is a listener process ready to take client commands, and that will re-start the listener process if it were to go down for any reason. This daemon process is a minimal process, taking up very little CPU and memory from your system, thereby providing the best possible environment for your requests to be handled quickly and efficiently. The daemon only stops restarting the listener once the listener informs the daemon that there was a normal shutdown.

The following graphic illustrates the architecture. Within the OPeNDAP configuration file there is an option to tell the server whether to run in single mode, allowing only one client to connect at a time, or in multiple mode, allowing multiple clients to connect to the server at the same time. In case this option `OPeNDAP.ProcessManagerMethod`, is set to single:



If the configuration option is set to multiple:



At this time there are not provisions for how many clients can connect at the same time, but it is a desirable feature to be able to tell the listener how many managers are available at a given time.

2. Starting and Stopping the OPeNDAP Server

First of all, let's assume that you have installed the software in a directory pointed to by the environment variable `OPENDAP_ROOT`. The client and server applications, along with the initialization file, will be installed into `${OPENDAP_ROOT}/bin`, and any help files will be installed in the directory `${OPENDAP_ROOT}/info`.

Before you can start the server you must first set the environment variable `OPENDAP_INI` to point to the initialization file used by the OPeNDAP server. (for `tcsh` syntax and that the file name is `opendap.ini`)

```
setenv OPENDAP_INI ${OPENDAP_ROOT}/bin/opendap.ini
```

for `sh` and `ksh` users with a file called `opendap.ini`:

```
OPENDAP_INI=${OPENDAP_ROOT}/bin/opendap.ini  
export OPENDAP_INI
```

To start the server, run (assuming `tcsh` syntax):

```
$> ${OPENDAP_ROOT}/bin/opendapdaemon -p <port number>
```

Your port can be any valid integer port number available for users to use. This will be the port that the listener will use to listen for client connections.

If you wish to see debugging information printed in the terminal, start the daemon this way:

```
$> ${OPENDAP_ROOT}/bin/opendapdaemon -p <port number> -d
```

If you are having problems with your installation, or problems retrieving information from the server, we may need for you to run the server with the debug flag turned on, along with verbose logging information that can be turned on in the initialization file (discussed later.)

Signals are used to restart the listener and to shutdown the server. Signal `INT` will stop the system, while signal `USR1` will restart the listener. You may restart the listener for multiple reasons but the most common one is to force the system to reload the server configuration file after you have made a change to it.

3. Using the OPeNDAP Client

Now to start serving up some data. First, let's make sure everything is working properly. We will issue some simple commands to make sure that the client is talking to the server. First, start the command-line client:

```
% opendapcmdln -h <server_host_machine> -p<port_number>
```

The `server_host_machine` is the machine on which you started the server. If on the same machine that you could say `localhost` here. The `port_number` is the same `port_number` that you used when starting the server.

If you just use these options then you will start using the command line version of the client. There are other options, but we'll start here. From here you should get a prompt. Let's try a simple command (remember to terminate each command with a semicolon):

```
OPeNDAPClient> show status;
```

You should get a response showing the status of the server:

```
Listener boot time: MDT Thu Jun 9 14:12:22 2005
```

Try another one:

```
OPeNDAPClient> show help;
```

This one should show both the OPeNDAP-g server help file and your help information.

```
OPeNDAPClient> exit
```

This one will exit out of interactive mode.

Other commands available to the OPeNDAP Client program:

- u specifies the name of a Unix socket for connecting to the server.
- h specifies the name of a host for TCP/IP connection
- p specifies the port where the server is listening for TCP/IP connection.
- x makes the client execute a query and exit. This flag requires the -f flag.
- f sets the target file name for the return stream from the server.
- i sets the target file name for a sequence of input commands.
- t sets the timeout in seconds and is optional.
- d sets the client session for debugging and is optional.
- v forces the client to show the version and exit

Connection Flags: -u or -p -h are required to connect to a server and specify either a Unix socket or a TCP/IP socket.

Input/Output Flags: you can specify that the input is from the command line argument with the -x flag or that the input must be read from a file with the -i flag. If you specify either -x or -i you must specify the name of a file for the output stream of the server with the -f flag. If neither the -x nor the -i flags are specified then the client goes into interactive mode. To exit out of interactive mode just type 'exit' (without the quotes) at the OPeNDAPClient> prompt.

4. Client commands

Current commands available with OPeNDAP:

```
show help;
    * shows this help
show containers;
    * shows all containers currently defined
show definitions;
    * shows all definitions currently defined
show version;
    * shows the version of OPeNDAP and each data type served by this server
show process;
    * shows the process number of this application
show status;
    * shows the status of the server
show keys;
    * shows all key/value pairs defined in the OPeNDAP initialization file

set container values <symbolic_name>,<real_name>,<container_type>;
    * defines a symbolic name representing a data container, usually a file,
      to be used by definitions, described below
    * <real_name> is the full path to a data file
    * <container_type> is the type of data that is in the file. For netcdf files it is nc,
      for cedar it is cedar, etc...

define <def_name> as <container_list>
    [with <container_x>.constraint="<constraint>",<container_x>.attributes="<attribute_list>"]
    [aggregate by "<aggregation_command>"];

    * creates a definition using one or more containers, constraints for
      each of the containers, attributes to be retrieved from each
      container, and an aggregation command. Constraints, attributes, and
      aggregation are all optional.
    * <container_list> is a comma separated list of container names.
    * The constraints, attributes and aggregation command must be enclosed in quotes

get das|dds|dods|ddx for <definition_name> [return as <return_name>];

    * dds: request the data descriptor structure
    * das: request the data attributes
    * dods: request for the data stream, this output is an octec binary
      stream which requires the analysis by the client libdap library
    * if you create additional response types then those would be added to this list.
      For example, Cedar adds a tab response, so the user could say 'get tab for d1;'
```

commit <definition_name> [as <filename>];
 * commits a definition to a netcdf file using the constraints, attributes and aggregation within the definition
 * if <filename> is left off then the file is created in a temporary directory defined by the key in the initialization file OPeNDAP.Temporary.Directory.
 * A container is created pointing to this file with the name set to the <definition_name>

delete container <container_name>;
 * deletes the specified container
delete definition <definition_name>;
 * deletes the specified definition
delete definitions;
 * deletes all of the currently defined definitions
there is no command "delete containers;"

** remember to terminate each command with a semicolon (;)

5. Configuration Options

Before we actually run the server, there's one more bit of configuration that you will need to do. The `opendap.ini` file is a file that contains key/value pairs used by the OPeNDAP-g server. For the most part you will need to make only two changes to this file.

The first is to the key `OPeNDAP.ServerAdministrator`. This should be set to the email address of the person who will be administrating this OPeNDAP-g server.

The other key is the `OPeNDAP.LogName` key, which defines the full path to the OPeNDAP log file. This log file will keep track of what requests are made and whether they are completed. The key right below this one, `OPeNDAP.LogVerbose`, defaults to no, meaning that the system won't dump debug information to the log file. If this key is set to yes then you will see much more information being dumped to the log file. This can be done in order to determine if there is a problem with the server and where it might be having that problem.

The other keys that are in the file are described below.

5.1 help files

There are two help files that contain information that is returned to a user who makes a 'show help' request. One key represents a plain text version of the file while the other represents an html version. The keys are `OPeNDAP.Help.TXT` and `OPeNDAP.Help.HTTP`. These two files are located in the `${OPENDAP_ROOT}/info` directory.

ESG also has a set of help files located in the `${OPENDAP_ROOT}/info` directory. They are `esg_help.txt` and `esg_help.html`. The keys for these files are `ESG.Help.TXT` and `ESG.Help.HTTP`.

5.2 buffering

Should the OPeNDAP-g server buffer requests for help, version and other informational requests? That's what this next key specifies. A value of no means that the information is not buffered and is transmitted as it is added to the informational response object.

`OPeNDAP.Info.Buffered=no`

5.3 Container persistence

What's this you ask? A container represents a set of data. For example, a NetCDF file is a container of data. For the most part, a container will represent a file. When defining a request you specify what containers will be used for that request using a container's symbolic name. What if, when defining a request, the user specifies a container that does not exist? This key defines what to do. The key `OPeNDAP.Container.Persistence` can be set to nice, meaning just log a message to the OPeNDAP log file, or to strict, which

means that the system should throw an error message and stop processing the request. The default set in the initialization file is nice.

5.4 Controlling memory usage and supporting exceptions dynamic memory requirements.

The key OPeNDAP.Memory.GlobalArea.EmergencyPoolSize tells the OPeNDAP-g server how much memory to reserve for memory exceptions. The fact is that throwing C++ exceptions when memory is exhausted required yet more dynamically allocated memory so this way we have a small memory pool available for exceptions to work if memory ever does run out. The value should be an integer which represents the number of MB to reserve.

The key OPeNDAP.Memory.GlobalArea.ControlHeap has possible values of {yes,no}. This key tells the OPeNDAP-g server whether or not you wish to control the maximum heap expansion for an OPeNDAP server processes. In almost 100% of Unix systems, allocating memory expands the heap (if necessary) so even when you free the memory (no leaks) the heap does not get contracted because free marks your memory blocks as available for your process but it does not return such memory to the OS. You can imagine that OPeNDAP-g server process serving a huge amount of data can try to expand its heap to the point of creating process starvation (assuming the process stays alive after handling the request) to other processes running on the machine. It is suggested to set this value to 'yes' if you expect heavy loads, otherwise keep it set to 'no'.

If you set the key OPeNDAP.Memory.GlobalArea.ControlHeap to 'yes', then you will need to set the key OPeNDAP.Memory.GlobalArea.MaximunHeapSize. This key tells the OPeNDAP server to expand the process heap to a predetermined value upon initialization. This size remains constant and works in two ways; it assures the server will not take all the memory in your machine and it assures that there is enough memory to work properly. Depending on the size of your containers and the size of the queries you expect to serve (as well as how much memory is available on your machine) you can set this value to an integer between 10 to 400, representing MB.

The key OPeNDAP.Memory.GlobalArea.Verbose has possible values {yes,no}. This key tells the OPeNDAP-g server to write extra debugging information in the log so developers and administrators can track memory management.

5.5 Single or multiple client connections at a time

As mentioned in the section OPeNDAP-g server architecture above, the OPeNDAP-g server can process a single client connect at a time or multiple client connections at a time. The key OPeNDAP.ProcessManagerMethod can be set to either single, which means that only one client can connect at any given time or to multiple, which means that multiple clients can connect to the server at any one time.

5.6 Connecting using UNIX sockets

A client can connect to your server either by specifying a port on the server machine or by using a UNIX socket. The key OPeNDAP.ServerUnixSocket should be set to the full path of your UNIX socket file. The default is /tmp/opendap.socket, which should work fine.

5.7 Connecting with a Browser

The key OPeNDAP.DefaultResponseMethod is for internal use only and should not be set by the user.

5.8 Directory for temporary files

The key OPeNDAP.Temporary.Directory is used to define where the OPeNDAP-g server will place temporary files, such as when the user uses the commit command. The default is /tmp.